DEEP LEARNING COM TENSORFLOW

SEQUÊNCIAS

DIEGO RODRIGUES DSC

INFNET

CRONOGRAMA

Dia	Aula	Trab
~~02/09~~	~~Workshop de Deep Learning~~	
~~04/09~~	~~Deep FeedForward~~	
~~09/09~~	~~Rede Neural Convolutiva~~	~~Modelo Baseline~~
~~11/09~~	~~AutoEncoders~~	
16/09	~~Representation & Transfer Learning~~	~~Modelo Profundo~~
18/09	Sequências	
23/09	Modelos Generativos	Deployment
25/09	Apresentação dos Trabalhos Parte II	

SEQUÊNCIAS

- PARTE 1 : TEORIA
  - BUSINESS UNDERSTANDING
    - ESTADO
    - FILTRO DE KALMAN
  - MODELING
    - REDES NEURAIS RECORRENTES
    - PROBLEMA DO GRADIENTE

- PARTE 2 : PRÁTICA
  - SEQUENCIAS
- PARTE 3 : TRABALHOS

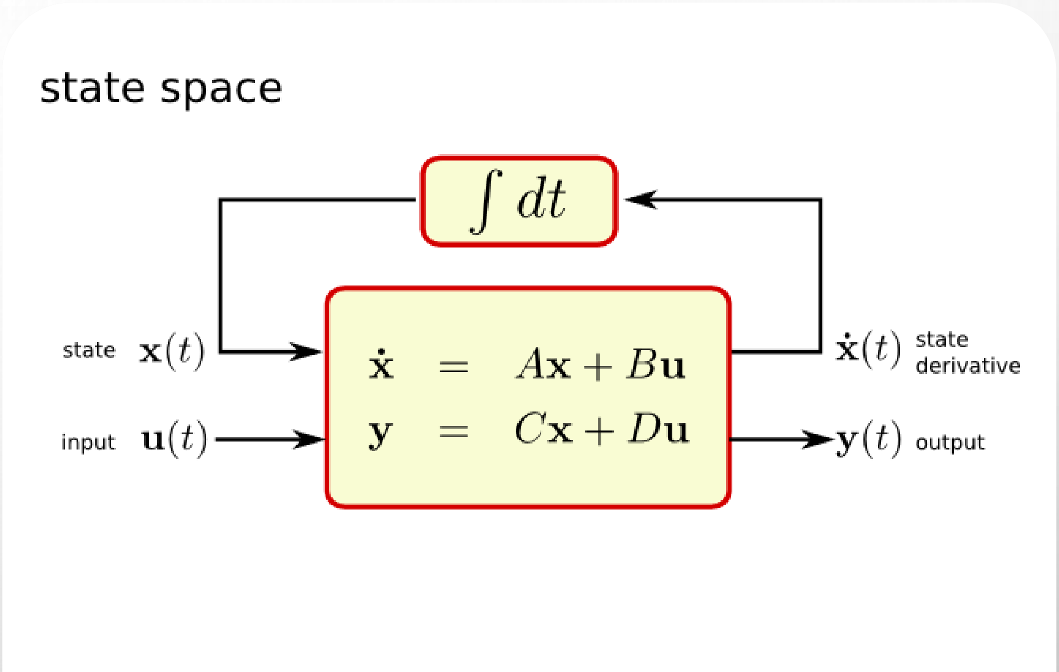
PARTE 1 : TEORIA

BUSINESS UNDERSTANDING

ESTADO

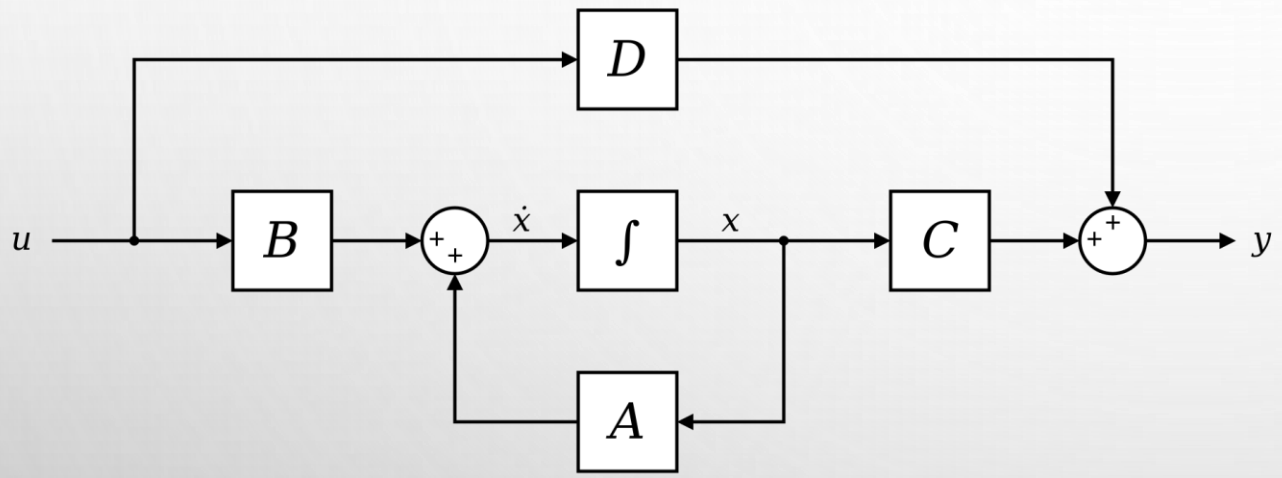
O conceito de estado se refere a um **conjunto de variáveis** que **capturam todas as informações necessárias** para **descrever completamente o status atual de um sistema.** O estado contém **todas as informações sobre o passado** do sistema que são **relevantes para prever seu comportamento futuro.**

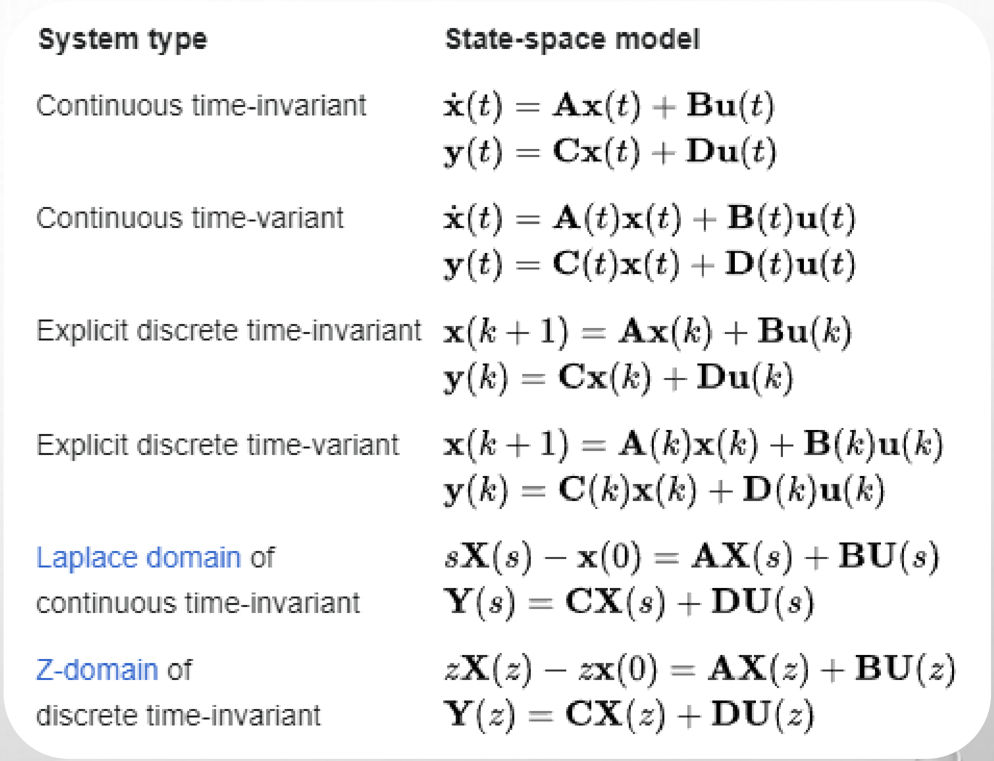
Em termos práticos, **controlar um sistema envolve manipular as entradas para direcioná-lo a um estado desejado,** respondendo adequadamente **às perturbações e mudanças no ambiente. Entender e modelar o estado** é fundamental para projetar controladores eficientes e para a estabilidade e desempenho do sistema.


state space
$\int dt$
state $\mathbf{x}(t)$
input $\mathbf{u}(t)$
$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$
$\mathbf{y} = C\mathbf{x} + D\mathbf{u}$
$\dot{\mathbf{x}}(t)$ state derivative
$\mathbf{y}(t)$ output

System represented as a collection of coupled linear first-order differential equations.

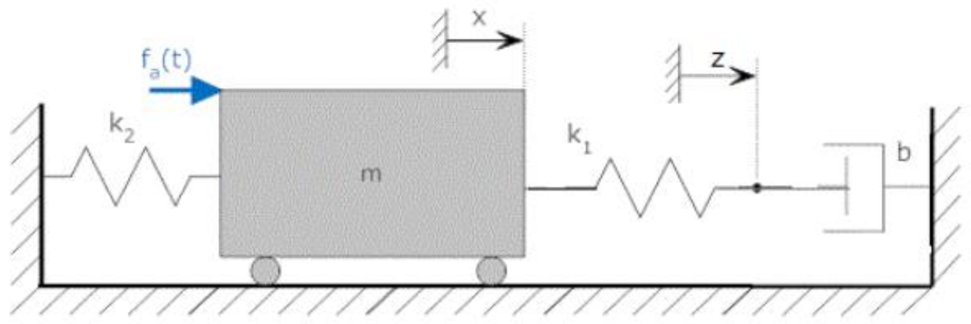
ESPAÇO DE ESTADOS



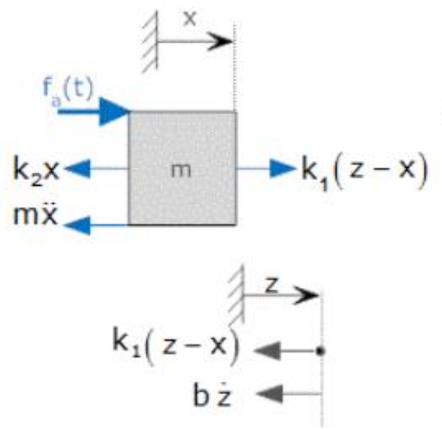


System type	State-space model
Continuous time-invariant	$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$ $\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$
Continuous time-variant	$\dot{\mathbf{x}}(t) = \mathbf{A}(t)\mathbf{x}(t) + \mathbf{B}(t)\mathbf{u}(t)$ $\mathbf{y}(t) = \mathbf{C}(t)\mathbf{x}(t) + \mathbf{D}(t)\mathbf{u}(t)$
Explicit discrete time-invariant	$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k)$ $\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k)$
Explicit discrete time-variant	$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{B}(k)\mathbf{u}(k)$ $\mathbf{y}(k) = \mathbf{C}(k)\mathbf{x}(k) + \mathbf{D}(k)\mathbf{u}(k)$
Laplace domain of continuous time-invariant	$s\mathbf{X}(s) - \mathbf{x}(0) = \mathbf{A}\mathbf{X}(s) + \mathbf{B}\mathbf{U}(s)$ $\mathbf{Y}(s) = \mathbf{C}\mathbf{X}(s) + \mathbf{D}\mathbf{U}(s)$
Z-domain of discrete time-invariant	$z\mathbf{X}(z) - z\mathbf{x}(0) = \mathbf{A}\mathbf{X}(z) + \mathbf{B}\mathbf{U}(z)$ $\mathbf{Y}(z) = \mathbf{C}\mathbf{X}(z) + \mathbf{D}\mathbf{U}(z)$

SISTEMA 1



Freebody Diagram | **Equation**



$$m\cdot\ddot{x}+k_1\cdot x+k_2x-k_1\cdot z=f_a$$

$$b\cdot\dot{z}+k_1z-k_1\cdot x=0$$

$$\dot{q}_1=\dot{x}=q_2$$

$$\dot{q}_2=\ddot{x}=\frac{1}{m}(f_a-k_1x-k_2x+k_1z)$$

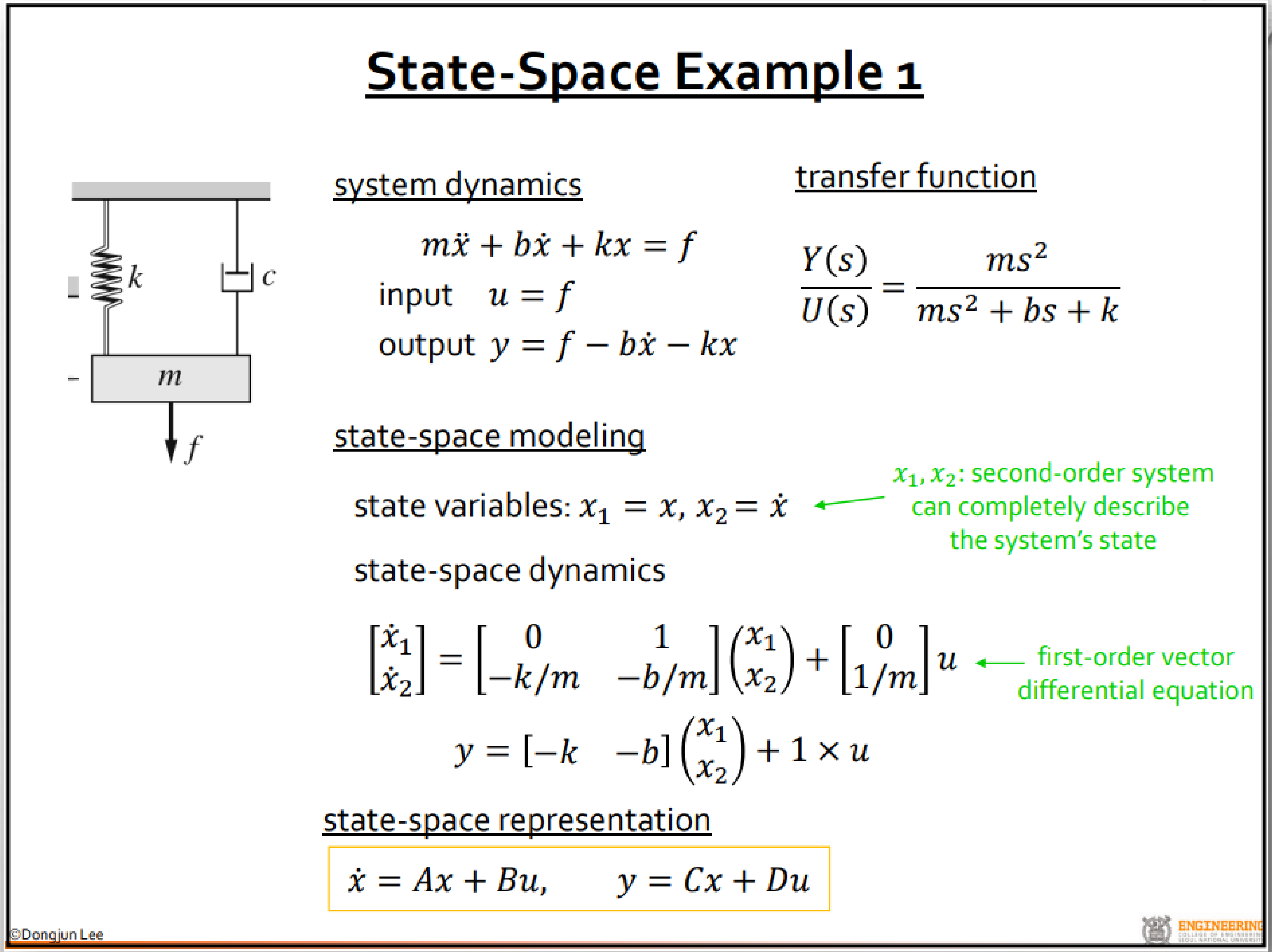
$$=\frac{1}{m}(f_a-k_1q_1-k_2q_1+k_1q_3)$$

$$\dot{q}_3=\dot{z}=\frac{k_1}{b}(x-z)=\frac{k_1}{b}(q_1-q_3)$$

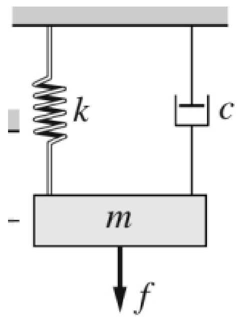
$$\dot{\mathbf{q}}=\mathbf{Aq}+\mathbf{B}u \qquad \mathbf{A}=\begin{bmatrix}0 & 1 & 0\\ -\frac{k_1+k_2}{m} & 0 & \frac{k_1}{m}\\ \frac{k_1}{b} & 0 & -\frac{k_1}{b}\end{bmatrix} \qquad \mathbf{B}=\begin{bmatrix}0\\ \frac{1}{m}\\ 0\end{bmatrix}$$

$$y=\mathbf{Cq}+Du \qquad \mathbf{C}=\begin{bmatrix}0 & 0 & 1\end{bmatrix} \qquad D=0$$

SISTEMA 2



State-Space Example 1



system dynamics

$$m\ddot{x} + b\dot{x} + kx = f$$

input $u = f$

output $y = f - b\dot{x} - kx$

transfer function

$$\frac{Y(s)}{U(s)} = \frac{ms^2}{ms^2 + bs + k}$$

state-space modeling

state variables: $x_1 = x,\ x_2 = \dot{x}$ ← x_1, x_2: second-order system can completely describe the system's state

state-space dynamics

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -k/m & -b/m\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + \begin{bmatrix}0\\ 1/m\end{bmatrix}u$$

← first-order vector differential equation

$$y = \begin{bmatrix}-k & -b\end{bmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + 1 \times u$$

state-space representation

$$\boxed{\dot{x} = Ax + Bu, \qquad y = Cx + Du}$$

https://ocw.snu.ac.kr/sites/default/files/NOTE/lecture%2012-14%20State%20Space_0.pdf

SISTEMA 3

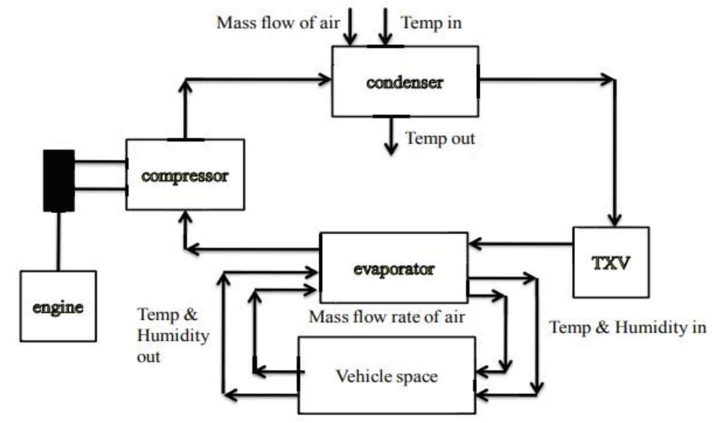


Figure 1. Automotive Air Conditioning System Block Diagram Representation

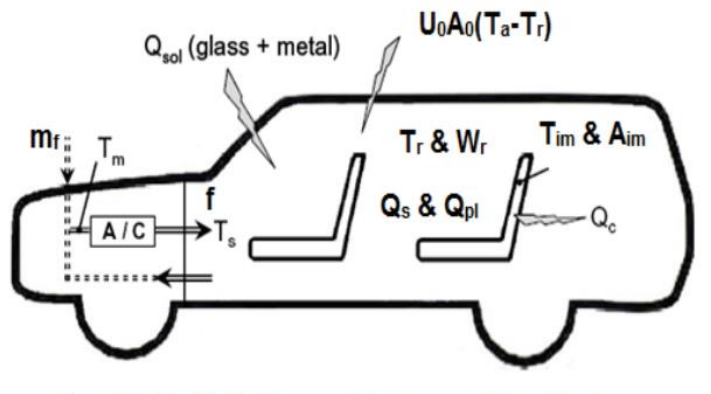


Figure 2. Physical Model of Passengers' Compartment and Thermal Loads

$$h_1(x,u) = \begin{bmatrix} -C_{pe}\rho f(T_m - T_s) + U_0A_0(T_a - T_r) + C_{pa}\rho f(T_a - T_r) + K_{spl}\,f \\ -h_{im}A_{im}(T_{im} - T_r) \\ C_p\rho f(T_d - T_s) + \propto_1 A_1\left(T_w - \frac{T_m+T_d}{2}\right) \\ C_p\rho f(T_d - T_s) + \rho f h_{fg}(W_m - W_s) + \propto_2 A_2\left(T_w - \frac{T_d+T_s}{2}\right) \\ \propto_1 A_1\left(\frac{T_m+T_d}{2} - T_w\right) + \propto_2 A_2\left(\frac{T_d+T_s}{2} - T_w\right) - M_{ref}(h_{r2} - h_{r1}) \\ -\rho f h_{fg}(W_m - W_s) + M_f h_{fg}(W_a - W_r) + Q_{pl} \\ 0 \end{bmatrix} \text{ - (14.a), } h_2(z) = \begin{bmatrix} Q_s + Q_{ps} \\ 0 \\ 0 \\ 0 \\ 0 \\ Q_{pl} \\ 0 \end{bmatrix} \text{ - (14.b)}$$

$$D = \begin{bmatrix} M_rC_{pr} & M_{im}C_c & 0 & 0 & 0 & 0 & 0 \\ 0 & M_{im}C_c & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & C_p\rho V_{h1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & C_p\rho V_{h2} & 0 & 0 & \rho V_{h2}h_{fg} \\ 0 & 0 & 0 & 0 & (C_p\rho V)_w & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & M_rh_{fg} & 0 \\ 0 & 0 & 0 & \left(\frac{-2\times 0.0198T_s+0.085}{1000}\right) & 0 & 0 & 1 \end{bmatrix} \text{ - (14.c)}$$

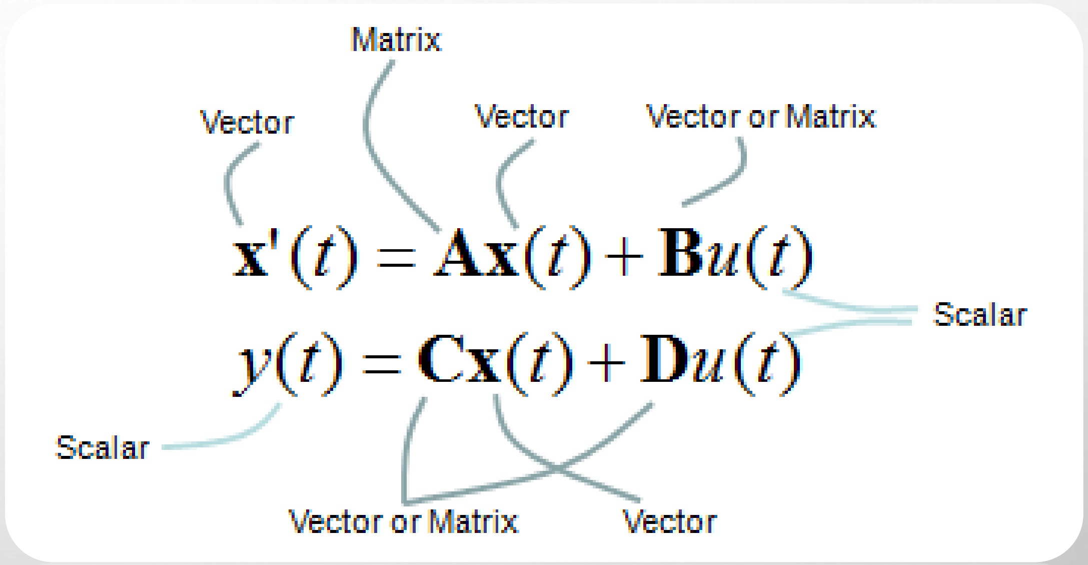
https://core.ac.uk/download/pdf/4955569.pdf

FILTRO DE KALMAN

O **Filtro de Kalman** é um algoritmo recursivo **que fornece estimativas de estados ocultos** de um sistema dinâmico com base em medições ruidosas ao longo do tempo. Utilizado amplamente em controle de sistemas, navegação, robótica, e **previsão de séries temporais**.

Cenário: Desejamos **rastrear o estado de um sistema** (como a posição e velocidade de um objeto) que está sujeito a:

- Dinâmica do sistema (movimento).
- Ruído nos sensores que medem essas variáveis.
- **Exemplo:** Estimar a posição de um veículo usando um sensor GPS ruidoso.


Matrix
Vector
Vector
Vector or Matrix
$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}u(t)$
$y(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}u(t)$
Scalar
Scalar
Vector or Matrix
Vector

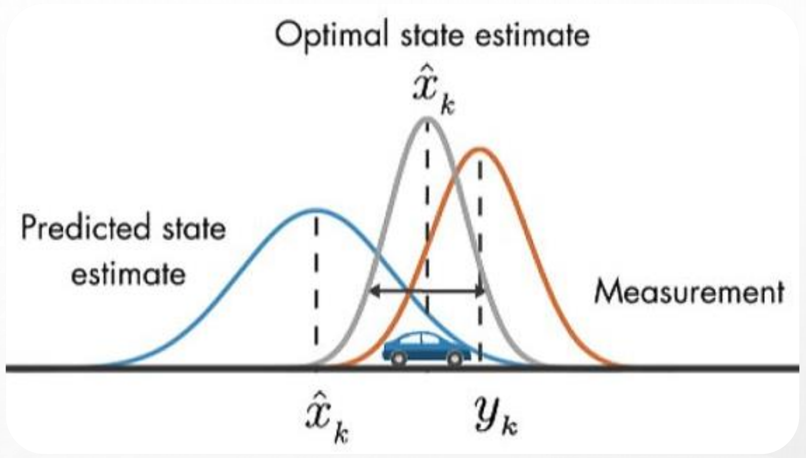
FILTRO DE KALMAN

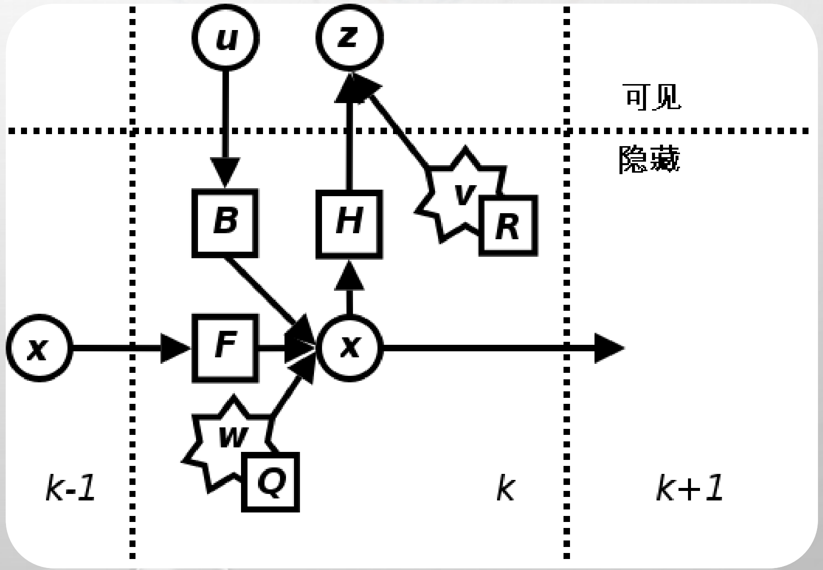
Predição

- Estima o próximo estado com base no estado atual e no modelo de transição.
- Atualiza a incerteza associada à predição.

Atualização

- Com base em novas observações, corrige a predição do estado.
- Calcula o ganho de Kalman (peso que indica a confiança na observação vs. a predição).
- Atualiza o estado estimado e reduz a incerteza.





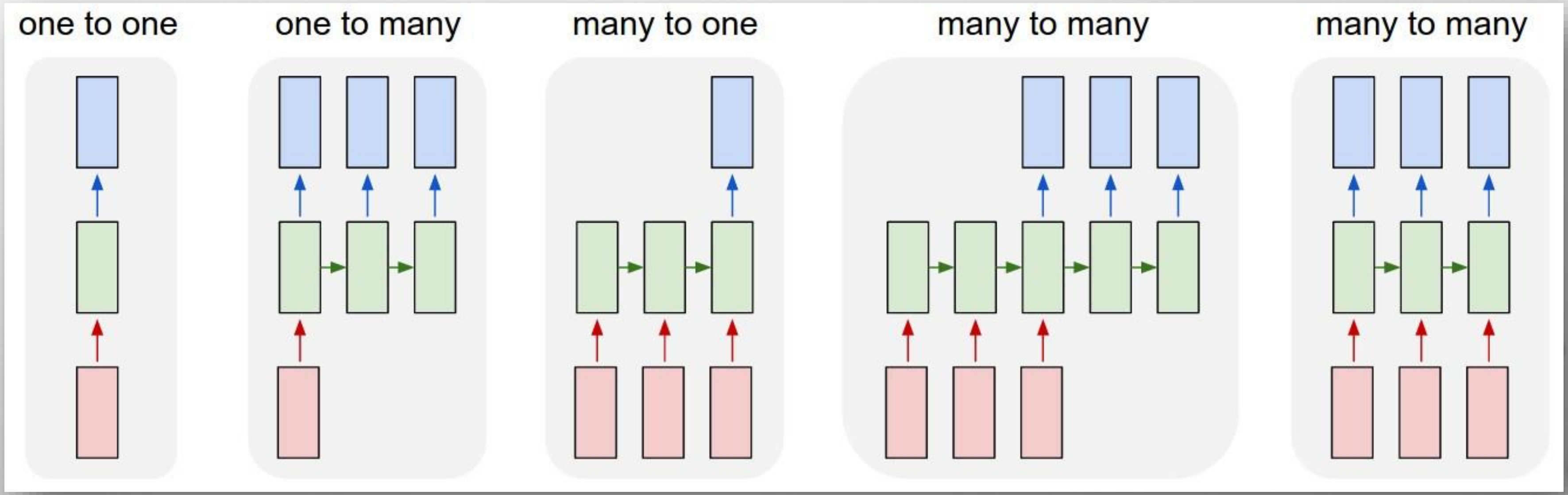
MODELING

APLICAÇÕES

Processamento de Linguagem Natural (NLP): Modelagem de dependências entre palavras em uma frase.

Previsão de Séries Temporais: Como em previsão de demanda de produtos, mercado financeiro, etc.

Reconhecimento de Fala: Mapeamento de uma sequência de áudio para texto.



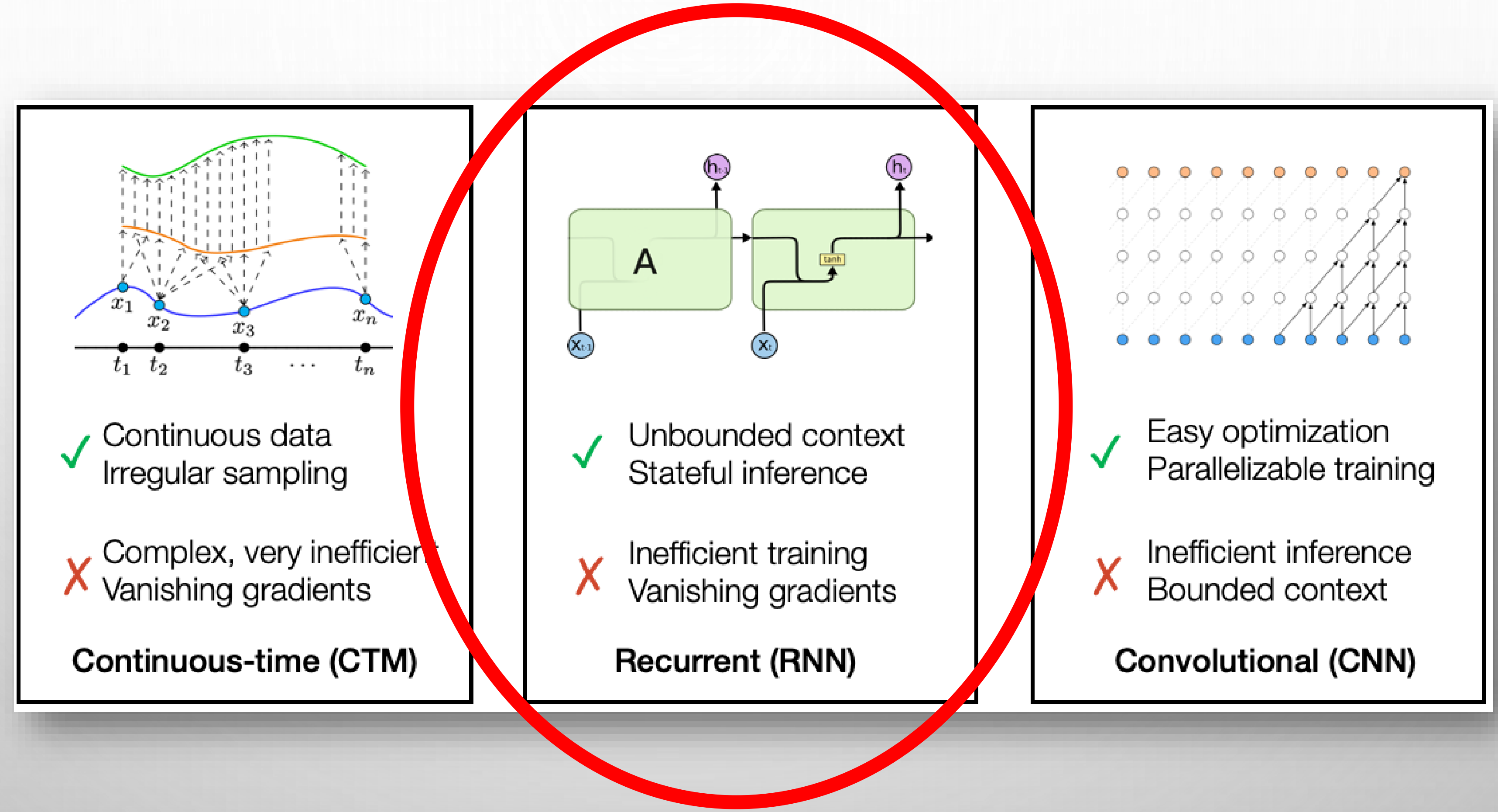
SÉRIES TEMPORAIS

$$y_t = T_t + C_t + S_t + I_t\,,$$

$$y_t = T_t \times C_t \times S_t \times I_t\,.$$

Tt : média / média móvel
Ct: Fourier Passa Baixas / Média Móvel
St: Fourier Passa Altas / Picos Remanescentes
Ii: auto-regressão / rede neural

SÉRIES TEMPORAIS COM REDES NEURAIS



✓ Continuous data
Irregular sampling

✗ Complex, very inefficien
Vanishing gradients

Continuous-time (CTM)

✓ Unbounded context
Stateful inference

✗ Inefficient training
Vanishing gradients

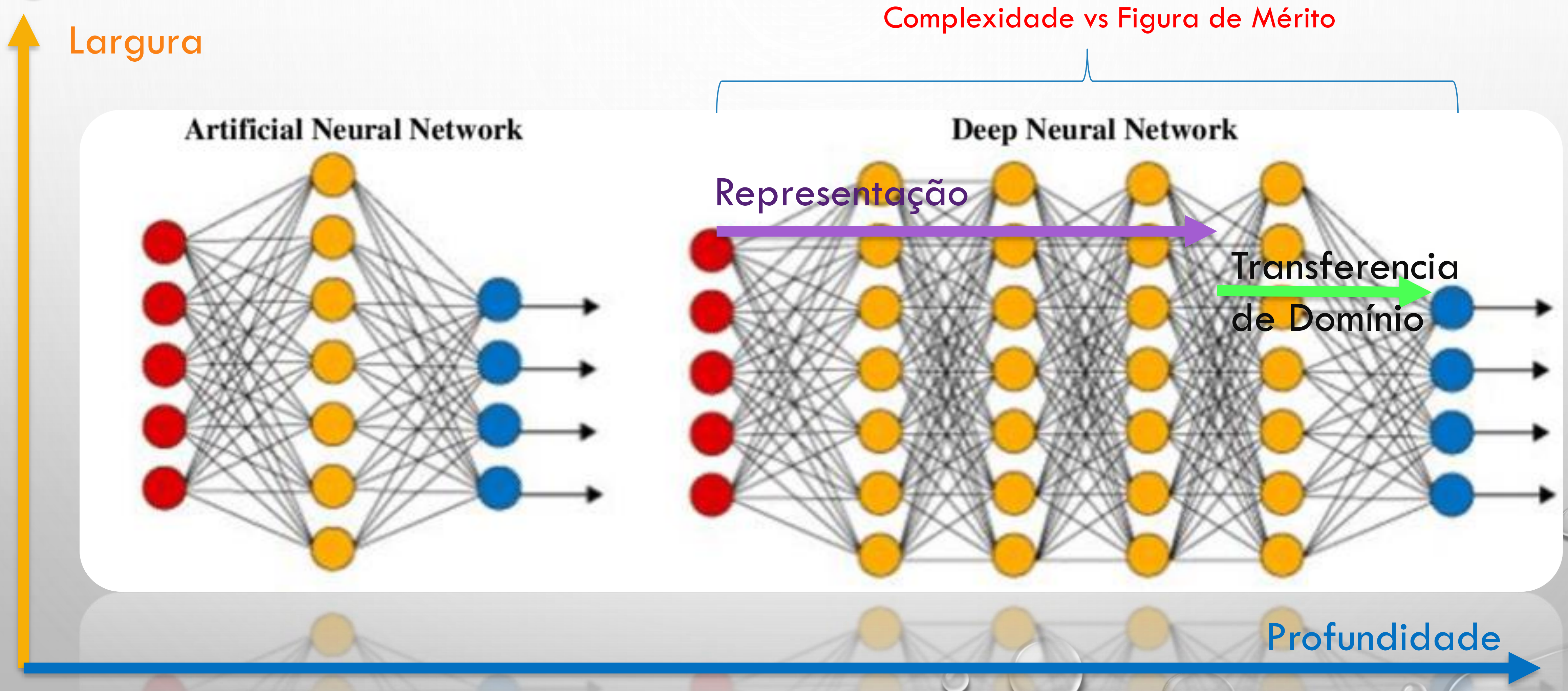
Recurrent (RNN)

✓ Easy optimization
Parallelizable training

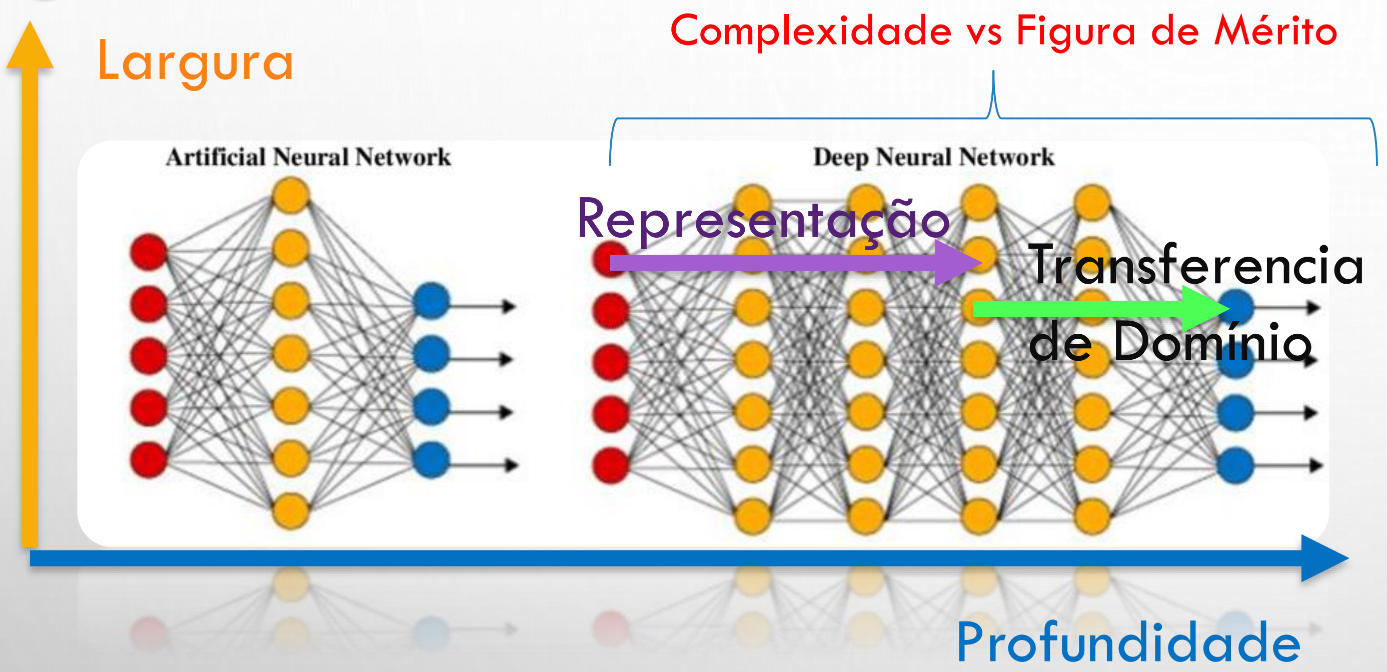
✗ Inefficient inference
Bounded context

Convolutional (CNN)

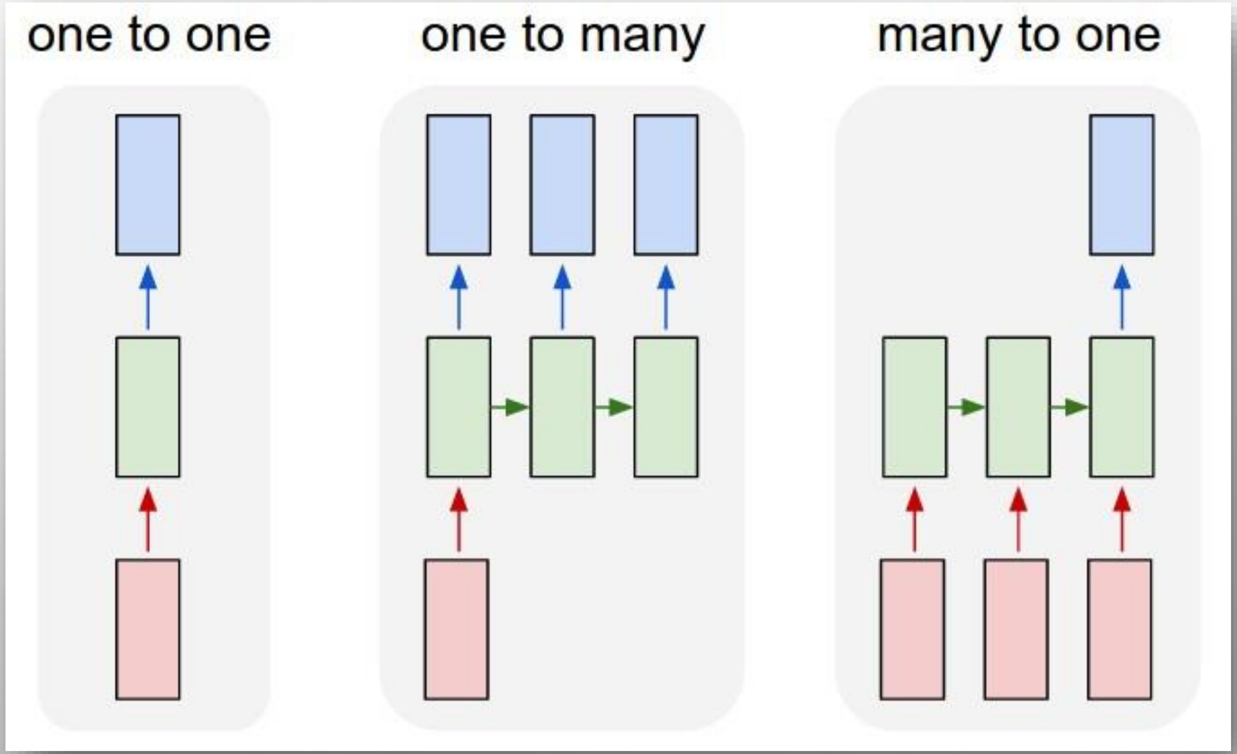
ESPAÇO DE REPRESENTAÇÕES

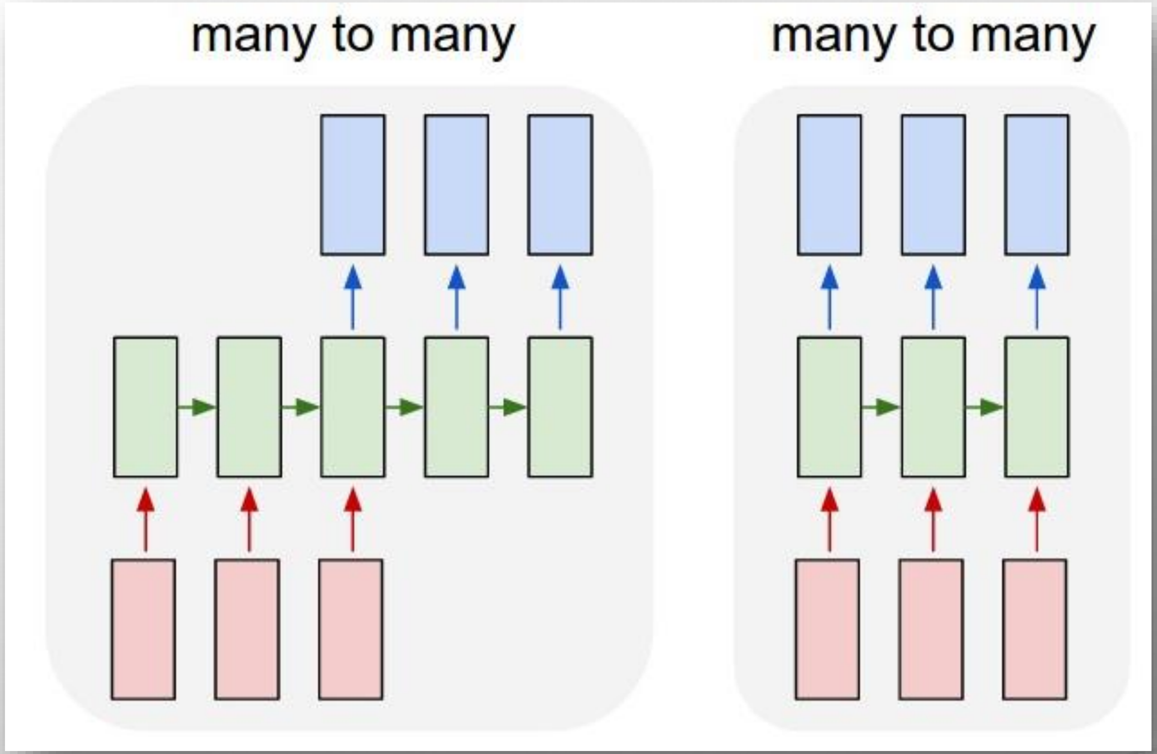


VERSÃO 1, VERSÃO 2, ETC



TIPOS DE JANELAMENTO





REDE NEURAL RECORRENTE

Redes Neurais Recorrentes são um tipo de rede neural projetada para lidar com **dados sequenciais.**

A **principal característica** de uma RNN é a **retroalimentação**: a **saída de uma unidade de tempo influencia a entrada da próxima unidade de tempo.**

Isso permite que a RNN **mantenha um estado interno** que **armazena informações sobre entradas anteriores,** tornando-a adequada para tarefas como **séries temporais, processamento de linguagem natural (NLP), e reconhecimento de fala.**

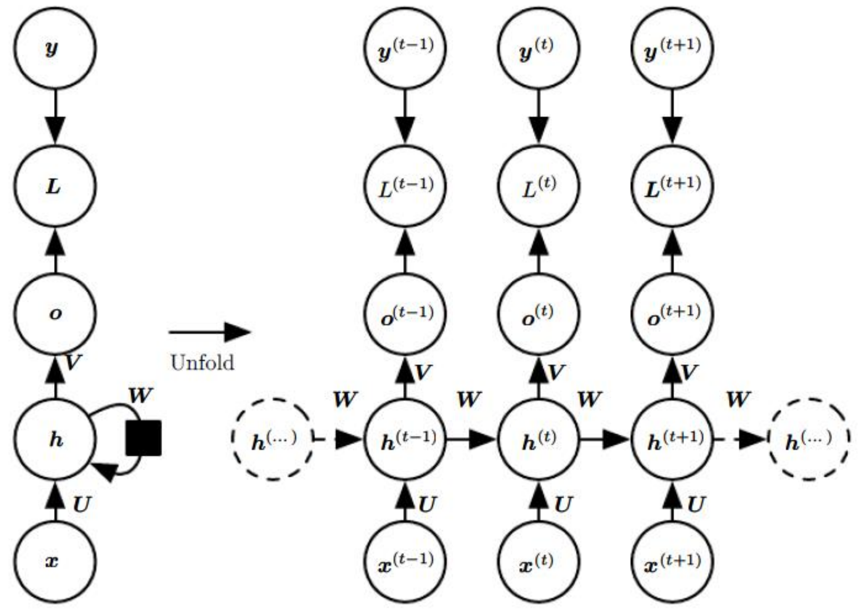


Figure 10.3: The computational graph to compute the training loss of a recurrent network that maps an input sequence of $\boldsymbol{x}$ values to a corresponding sequence of output $\boldsymbol{o}$ values. A loss L measures how far each $\boldsymbol{o}$ is from the corresponding training target $\boldsymbol{y}$. When using softmax outputs, we assume $\boldsymbol{o}$ is the unnormalized log probabilities. The loss L internally computes $\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{o})$ and compares this to the target $\boldsymbol{y}$. The RNN has input to hidden connections parametrized by a weight matrix $\boldsymbol{U}$, hidden-to-hidden recurrent connections parametrized by a weight matrix $\boldsymbol{W}$, and hidden-to-output connections parametrized by a weight matrix $\boldsymbol{V}$. Equation 10.8 defines forward propagation in this model. *(Left)*The RNN and its loss drawn with recurrent connections. *(Right)*The same seen as an time-unfolded computational graph, where each node is now associated with one particular time instance.

REDE NEURAL RECORRENTE II

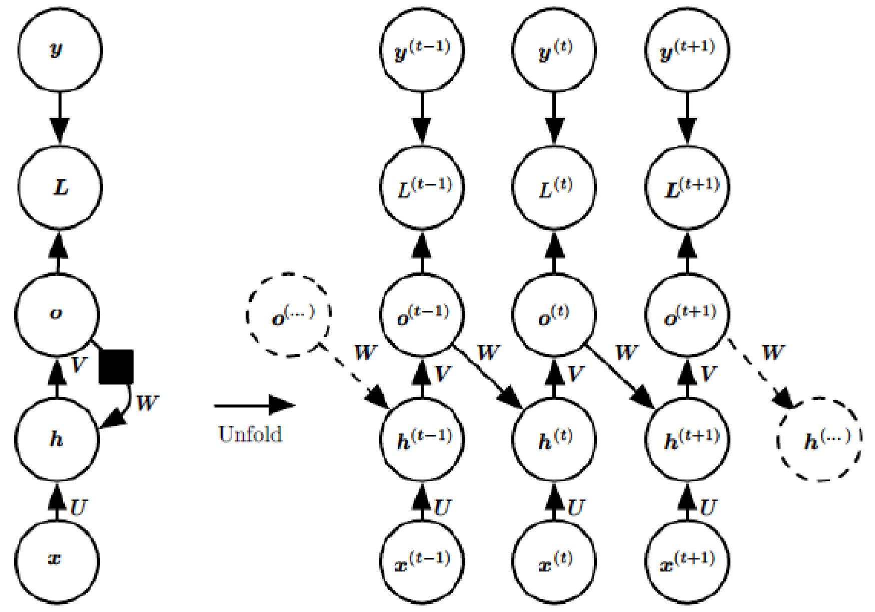


Figure 10.4: An RNN whose only recurrence is the feedback connection from the output to the hidden layer. At each time step t, the input is $\boldsymbol{x}_t$, the hidden layer activations are $\boldsymbol{h}^{(t)}$, the outputs are $\boldsymbol{o}^{(t)}$, the targets are $\boldsymbol{y}^{(t)}$ and the loss is $L^{(t)}$. *(Left)*Circuit diagram. *(Right)*Unfolded computational graph. Such an RNN is less powerful (can express a smaller set of functions) than those in the family represented by figure 10.3. The RNN in figure 10.3 can choose to put any information it wants about the past into its hidden representation $\boldsymbol{h}$ and transmit $\boldsymbol{h}$ to the future. The RNN in this figure is trained to put a specific output value into $\boldsymbol{o}$, and $\boldsymbol{o}$ is the only information it is allowed to send to the future. There are no direct connections from $\boldsymbol{h}$ going forward. The previous $\boldsymbol{h}$ is connected to the present only indirectly, via the predictions it was used to produce. Unless $\boldsymbol{o}$ is very high-dimensional and rich, it will usually lack important information from the past. This makes the RNN in this figure less powerful, but it may be easier to train because each time step can be trained in isolation from the others, allowing greater parallelization during training, as described in section 10.2.1.

REDE NEURAL RECORRENTE III

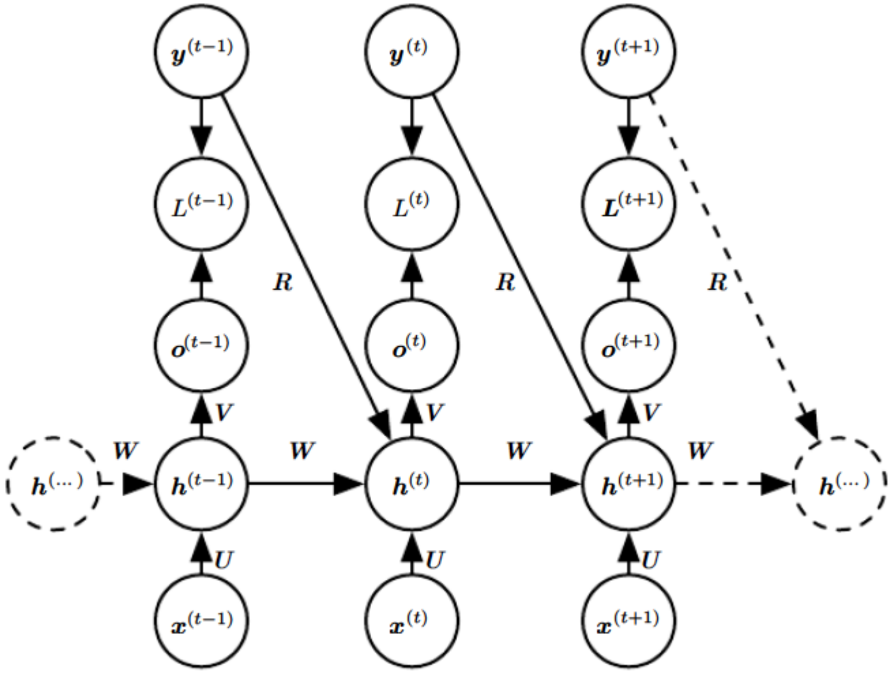
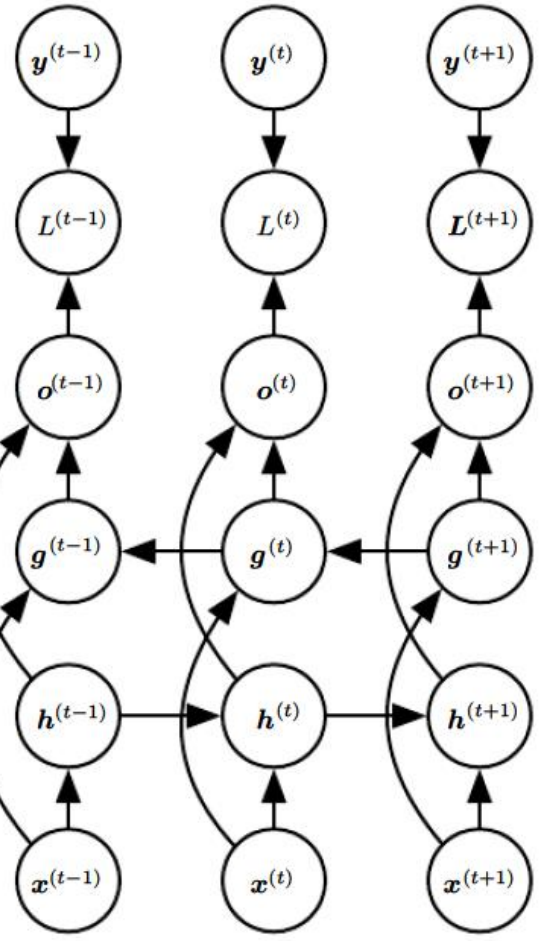


Figure 10.10: A conditional recurrent neural network mapping a variable-length sequence of $\boldsymbol{x}$ values into a distribution over sequences of $\boldsymbol{y}$ values of the same length. Compared to figure 10.3, this RNN contains connections from the previous output to the current state. These connections allow this RNN to model an arbitrary distribution over sequences of $\boldsymbol{y}$ given sequences of $\boldsymbol{x}$ of the same length. The RNN of figure 10.3 is only able to represent distributions in which the $\boldsymbol{y}$ values are conditionally independent from each other given the $\boldsymbol{x}$ values.

REDE NEURAL RECORRENTE BIDIRECIONAL



$$\begin{tikzcd}
\boldsymbol{y}^{(t-1)} \arrow[d] & \boldsymbol{y}^{(t)} \arrow[d] & \boldsymbol{y}^{(t+1)} \arrow[d] \\
L^{(t-1)} & L^{(t)} & L^{(t+1)} \\
\boldsymbol{o}^{(t-1)} \arrow[u] & \boldsymbol{o}^{(t)} \arrow[u] & \boldsymbol{o}^{(t+1)} \arrow[u] \\
\boldsymbol{g}^{(t-1)} \arrow[u] & \boldsymbol{g}^{(t)} \arrow[u] \arrow[l] & \boldsymbol{g}^{(t+1)} \arrow[u] \arrow[l] \\
\boldsymbol{h}^{(t-1)} \arrow[uu, bend left] \arrow[r] & \boldsymbol{h}^{(t)} \arrow[uu, bend left] \arrow[r] & \boldsymbol{h}^{(t+1)} \arrow[uu, bend left] \\
\boldsymbol{x}^{(t-1)} \arrow[u] \arrow[uu, bend left] & \boldsymbol{x}^{(t)} \arrow[u] \arrow[uu, bend left] & \boldsymbol{x}^{(t+1)} \arrow[u] \arrow[uu, bend left]
\end{tikzcd}$$

Figure 10.11: Computation of a typical bidirectional recurrent neural network, meant to learn to map input sequences $\boldsymbol{x}$ to target sequences $\boldsymbol{y}$, with loss $L^{(t)}$ at each step t. The $\boldsymbol{h}$ recurrence propagates information forward in time (towards the right) while the $\boldsymbol{g}$ recurrence propagates information backward in time (towards the left). Thus at each point t, the output units $\boldsymbol{o}^{(t)}$ can benefit from a relevant summary of the past in its $\boldsymbol{h}^{(t)}$ input and from a relevant summary of the future in its $\boldsymbol{g}^{(t)}$ input.

REDE NEURAL ENCODER RECORRENTE DECODER

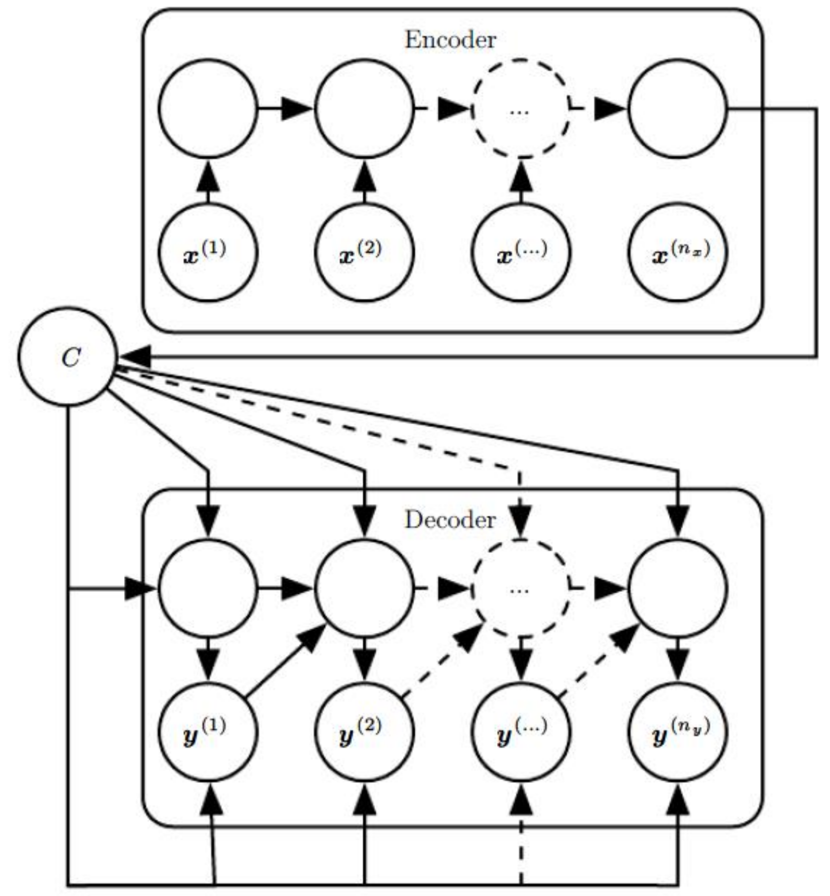


Figure 10.12: Example of an encoder-decoder or sequence-to-sequence RNN architecture, for learning to generate an output sequence $(\mathbf{y}^{(1)}, \dots, \mathbf{y}^{(n_y)})$ given an input sequence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(n_x)})$. It is composed of an encoder RNN that reads the input sequence and a decoder RNN that generates the output sequence (or computes the probability of a given output sequence). The final hidden state of the encoder RNN is used to compute a generally fixed-size context variable C which represents a semantic summary of the input sequence and is given as input to the decoder RNN.

REDE RECORRENTE PROFUNDA

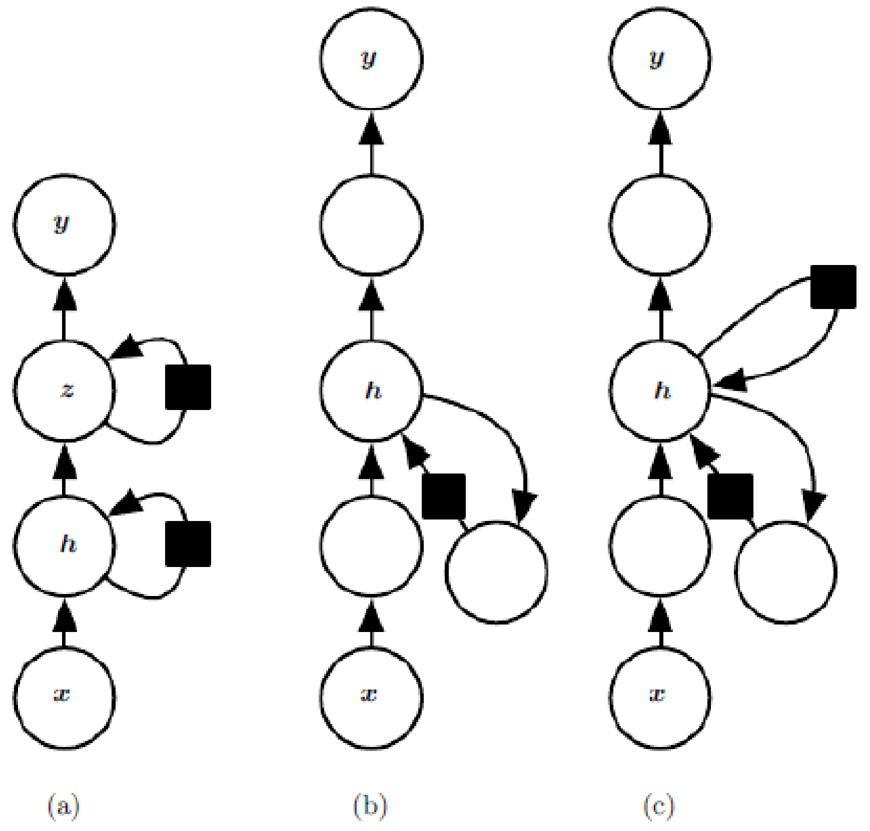


Figure 10.13: A recurrent neural network can be made deep in many ways (Pascanu *et al.*, 2014a). *(a)*The hidden recurrent state can be broken down into groups organized hierarchically. *(b)*Deeper computation (e.g., an MLP) can be introduced in the input-to-hidden, hidden-to-hidden and hidden-to-output parts. This may lengthen the shortest path linking different time steps. *(c)*The path-lengthening effect can be mitigated by introducing skip connections.

REDE RECORRENTE RECURSIVA

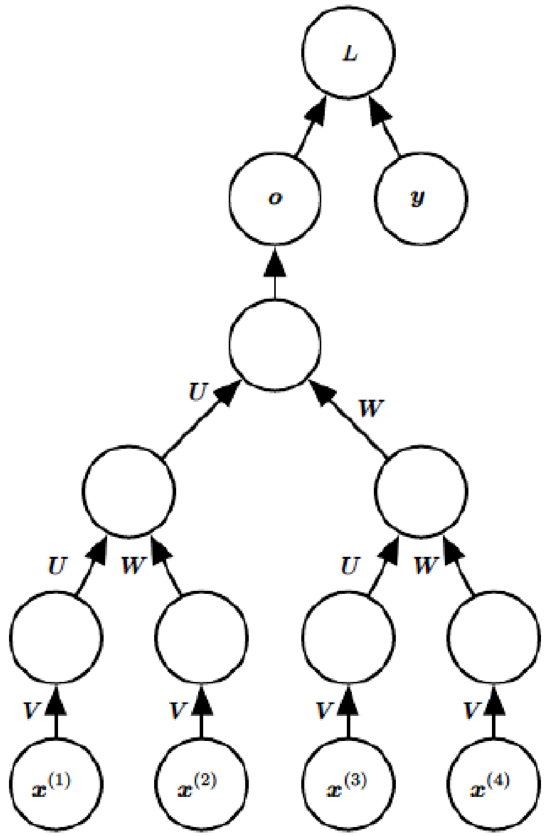
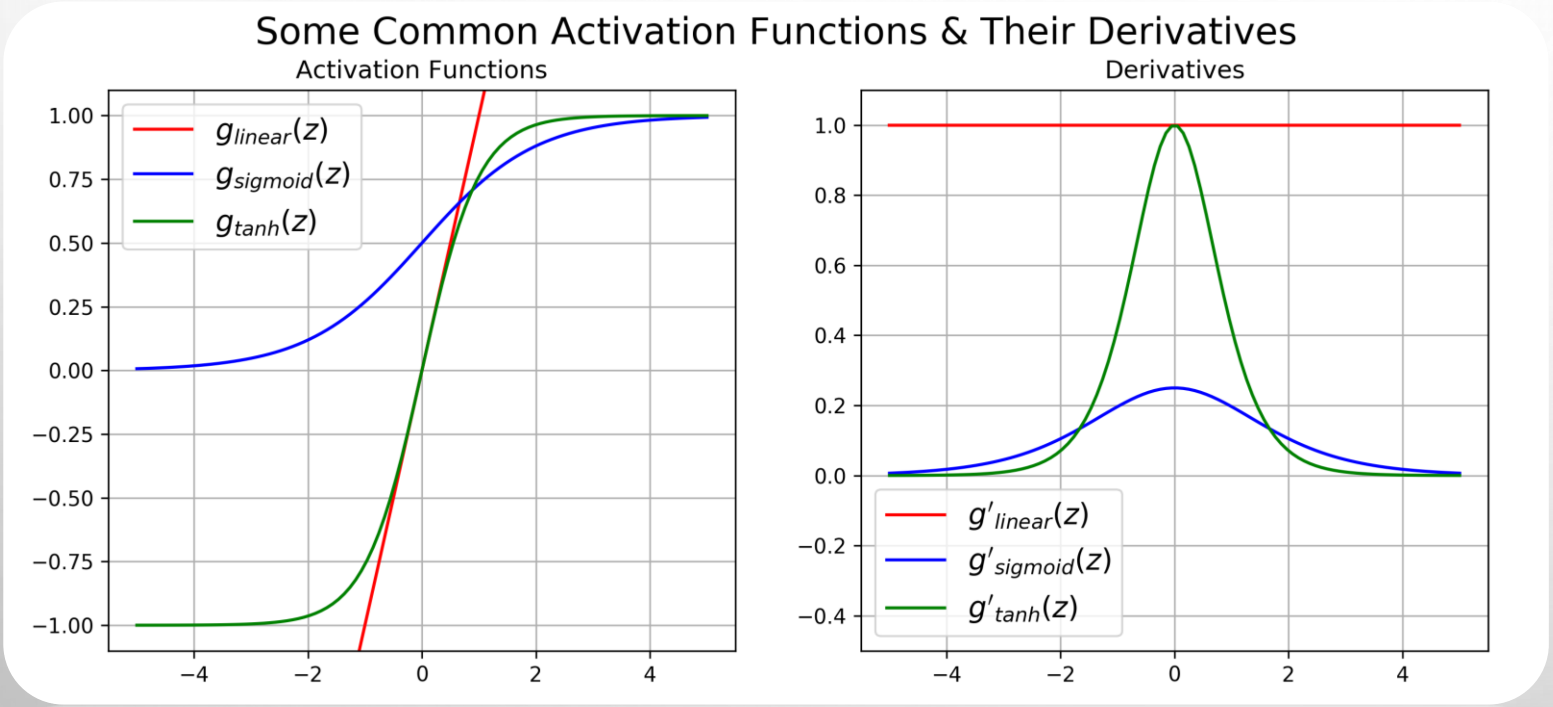
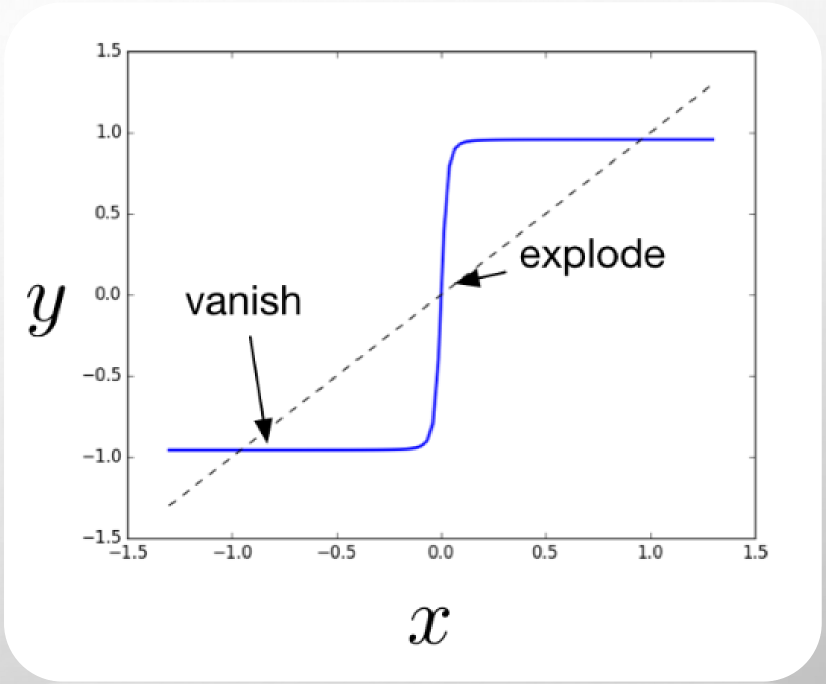


Figure 10.14: A recursive network has a computational graph that generalizes that of the recurrent network from a chain to a tree. A variable-size sequence $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(t)}$ can be mapped to a fixed-size representation (the output $\boldsymbol{o}$), with a fixed set of parameters (the weight matrices $\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{W}$). The figure illustrates a supervised learning case in which some target $\boldsymbol{y}$ is provided which is associated with the whole sequence.

O PROBLEMA DA DISSIPAÇÃO DO GRADIENTE





REVISÃO

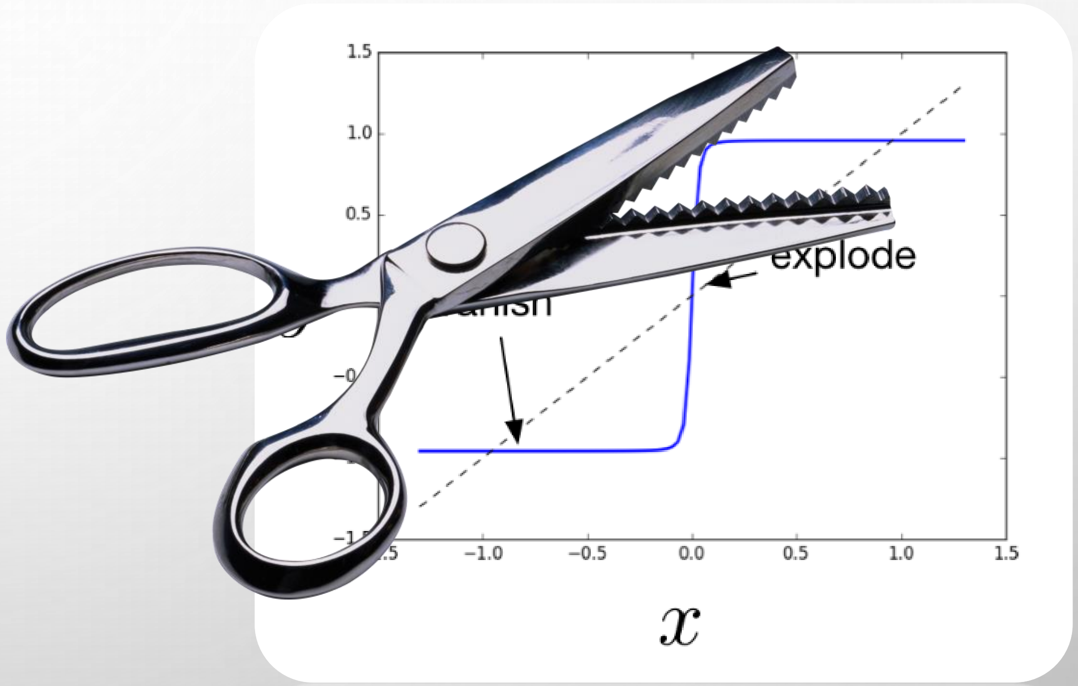
GRADIENT CLIPPING

O que é: Técnica para limitar a magnitude dos gradientes durante o treinamento de redes neurais.

Por que usar: Evita a explosão do gradiente em RNNs, melhorando a estabilidade do treinamento.

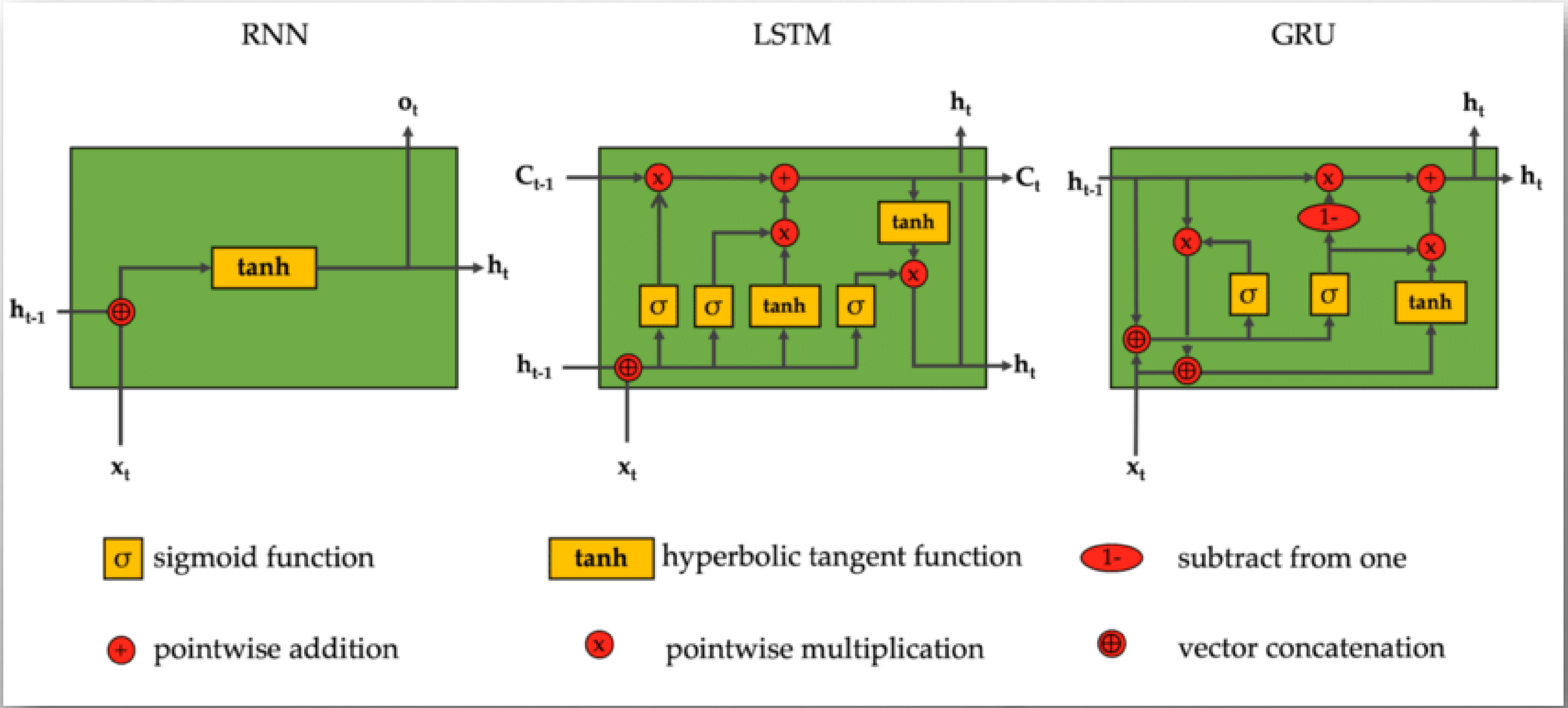
Como funciona:

- Clipping pela Norma (clipnorm): Limita a norma L2 do gradiente a um valor máximo.
- Clipping pelo Valor (clipvalue): Limita o valor absoluto de cada componente do gradiente.



REVISÃO

CONTROLE DO GRADIENTE



LSTM

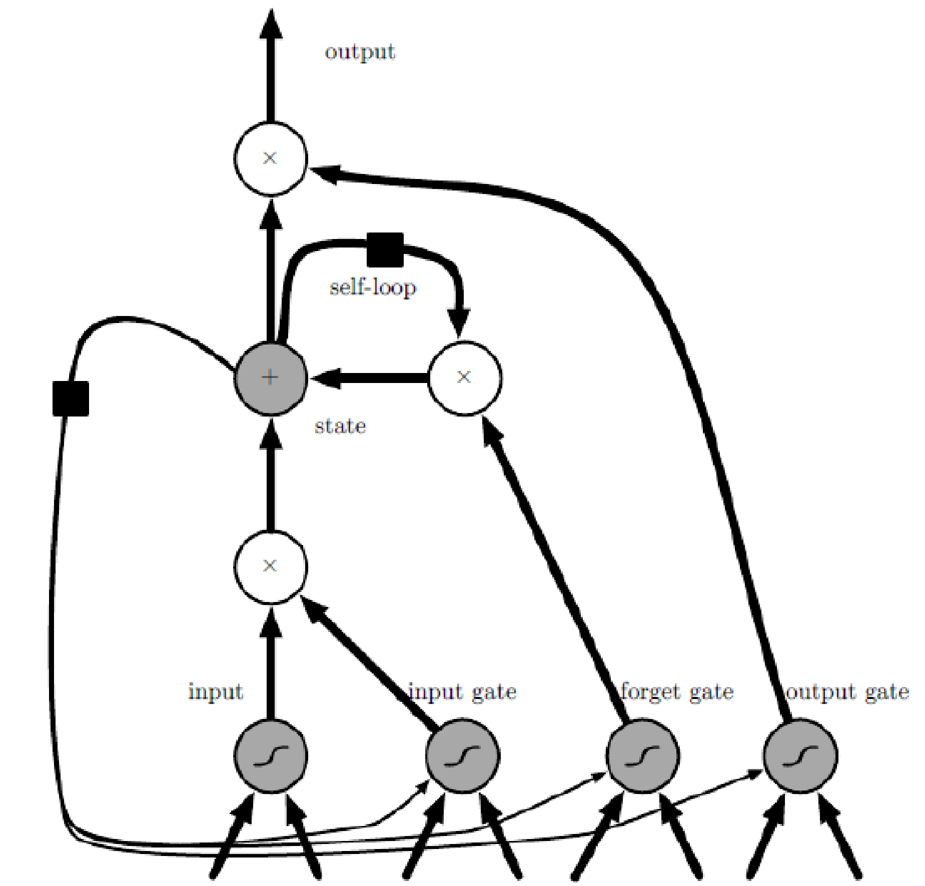
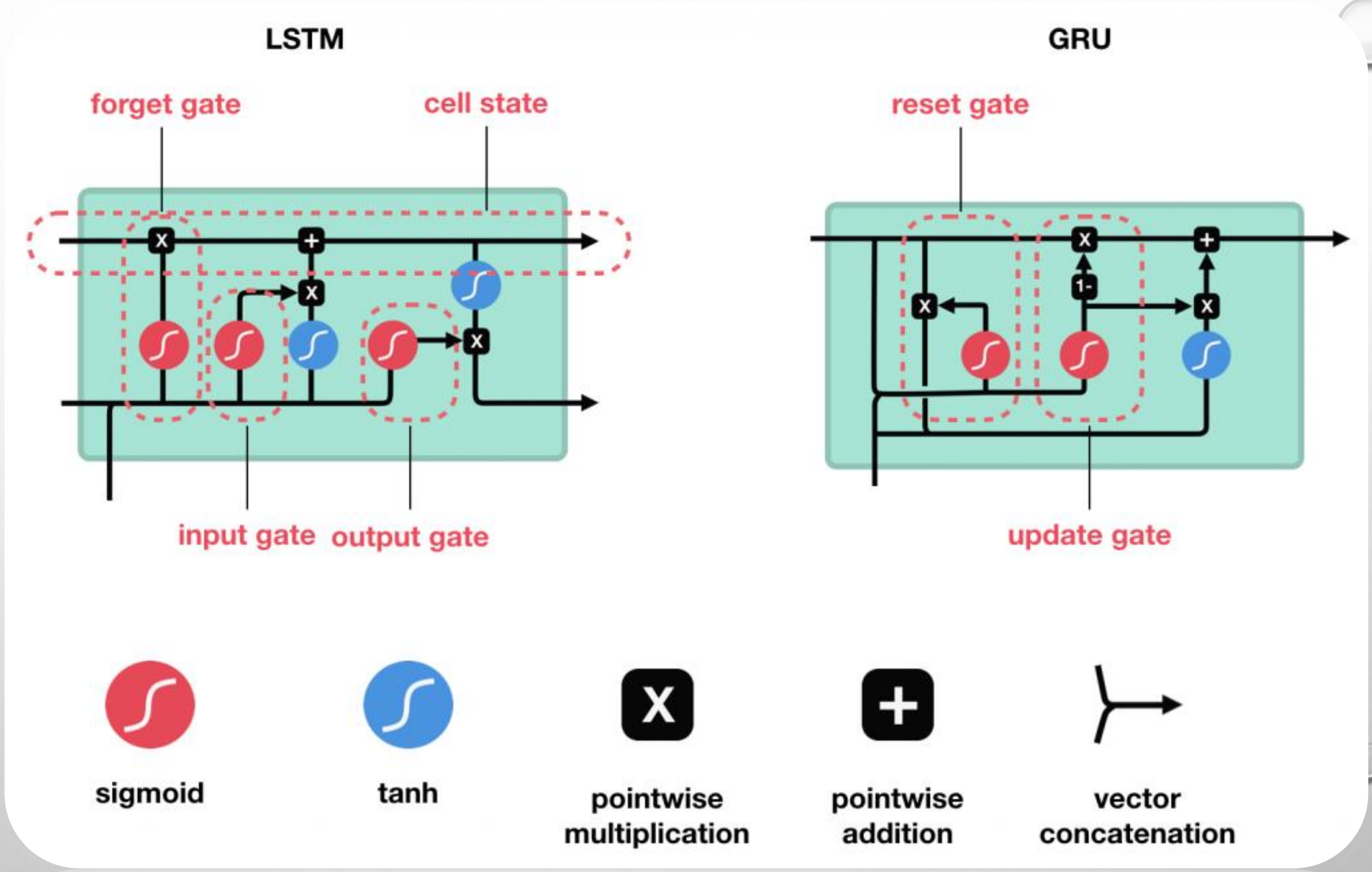


Figure 10.16: Block diagram of the LSTM recurrent network "cell." Cells are connected recurrently to each other, replacing the usual hidden units of ordinary recurrent networks. An input feature is computed with a regular artificial neuron unit. Its value can be accumulated into the state if the sigmoidal input gate allows it. The state unit has a linear self-loop whose weight is controlled by the forget gate. The output of the cell can be shut off by the output gate. All the gating units have a sigmoid nonlinearity, while the input unit can have any squashing nonlinearity. The state unit can also be used as an extra input to the gating units. The black square indicates a delay of a single time step.

GRU


LSTM
forget gate
cell state
input gate
output gate
GRU
reset gate
update gate
1-
sigmoid
tanh
pointwise multiplication
pointwise addition
vector concatenation

PARTE 2 : PRÁTICA

AMBIENTE PYTHON

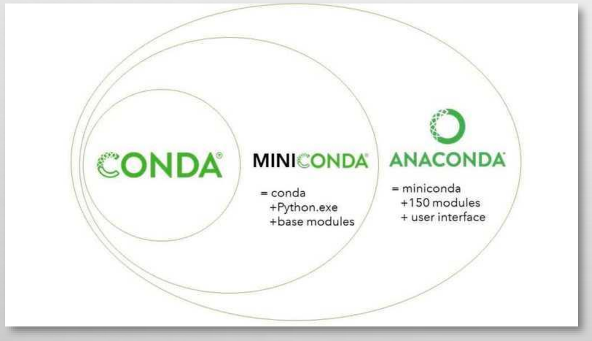
6. Machine
Learning

6. Deployment

5. Visualização

4. Variáveis
Aleatórias

1. Editor de Código



2. Gestor de Ambiente

3. Ambiente
Python do Projeto

3. Notebook
Dinâmico

WORKSHOP

- ~~**QUAL A TOPOLOGIA DE DEEP LEARNING ADEQUADA PARA O MEU TRABALHO?**~~
- ~~**QUAL CAPÍTULO DO LIVRO MELHOR SE ENQUADRA NO MEU TRABALHO?**~~
- ~~**AULA 3: NOVO CICLO DE BUSINESS UNDERSTANDING / GRUPO + MODELO BASELINE TREINADO**~~
- **AULA 5 OU 7: MODELO PROFUNDO TREINADO**
- **AULA 7: DEPLOYMENT DO MODELO***
- **AULA 3-7 > APRESENTAÇÃO TEÓRICA DA(S) TOPOLOGIA(S) + LEITURA DE ARTIGO + ACOMPANHAMENTO DOS TRABALHOS + DEEP DIVE NO CÓDIGO (POR GRUPO)**
- APRESENTAÇÃO FINAL DOS TRABALHOS

**Passo Opcional*